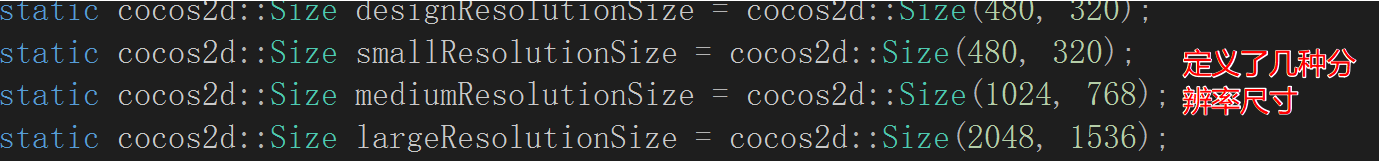
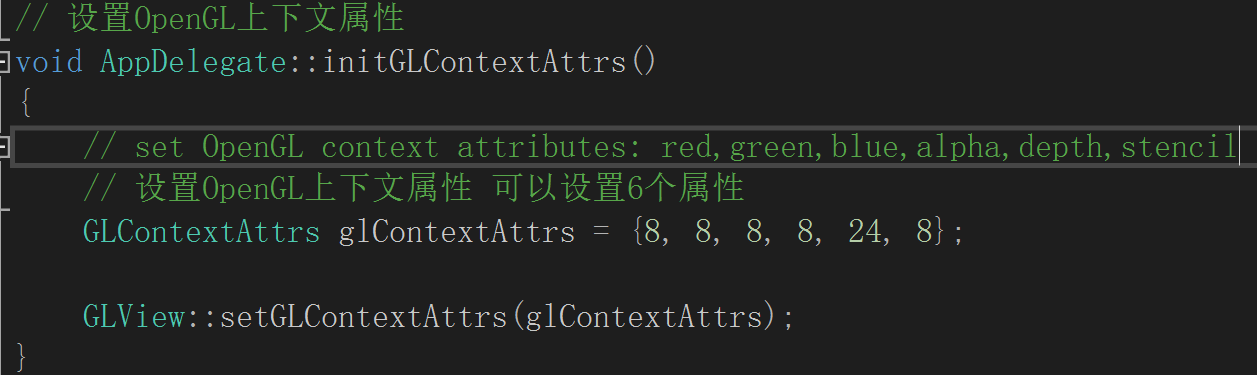
~Cocos2d-x 3.x 创建新工程命令

Python cocos new HelloWorld -p com.work1 -l cpp -d <工程生成目录>

~AppDelegate继承了cocos2d::Application,cocos2d::Application是Cocos2d-x引擎提供的基类



修改分辨率的话修改前两个







~循环调度

在HelloWorldSence.h中添加update(float dt);声明

virtual void update(float dt);

选取元素

auto label = Label::createWithTTF("Hello World", "fonts/Marker Felt.ttf", 24);

label->setTag(123);

更新函数

this->scheduleUpdate();

写函数体

void HelloWorld::update(float dt)

{

auto label = this->getChildByTag(123);

Label->setPosition(label->getPosition() + Vec2(2,-2));

}

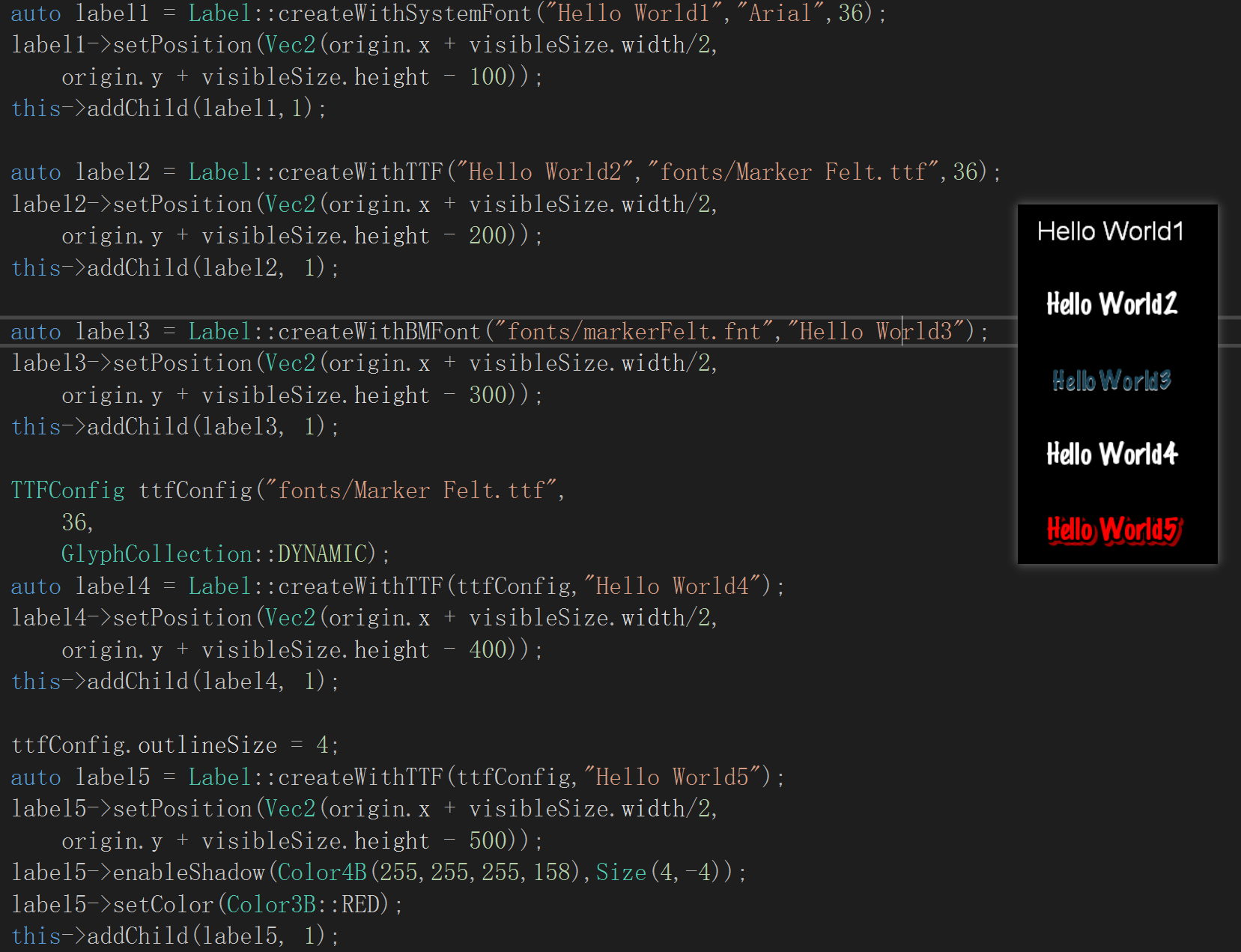
停止更新

void HelloWorld::menuCloseCallback(Ref\* pSender)

{

unscheduleUpdate();

~文字效果



~文字类型TTFConfig结构体定义

\_ttfConfig(const char \* filePath = "", //字体文件路径

int size = 12, //字体大小

const GlyphCollection&glyphCollection = GlyphCollection::DYNAMIC, //字体库类型

const char \* custonGlyphCollection = nullptr, //自定义字体库

bool useDistanceField = false, //开启字段距离字体开关

int outline = 0 //字体描边

)

~设定随机数

CCRANDOM\_0\_1()

~anchorPoint的计算公式

(w1 / w2, h1 / h2) 即左下角(0, 0) 右上角(1, 1)

~获得屏幕宽高

Size visibleSize = Director::getInstance()->getVisibleSize();

~vector数据结构

在HelloWorldScene.h中声明该结构

cocos2d::Vector<cocos2d::Sprite \* >list;

在HelloWorldScene.cpp的init()函数中构造 其中MAX\_COUNT为个数

this->list = Vector<Sprite \* >(MAX\_COUNT);

for (int i = 0; i < MAX\_COUNT; i++){

Sprite \* sprite = Sprite::create("Hello.png");

this->list.pushBack(sprite);

}

在void HelloWorld::menuCloseCallback(Ref\* pSender)中写按钮事件

Ref \* obj = nullptr;

log("List count = %d", this->list.size());

Size visibleSize = Director::getInstance()->getVisibleSize();

for (const auto&sprite : this->list){

int x = CCRANDOM\_0\_1() \* visibleSize.width;

int y = CCRANDOM\_0\_1() \* visibleSize.height;

sprite->setPosition(Vec2(x,y));

this->removeChild(sprite);

this->addChild(sprite);

}

完成构建随机出现多粒子按钮事件

~Map数据结构

在HelloWorldScene.h中声明该结构

cocos2d::Map<std::string, cocos2d::Sprite \* >map;

在HelloWorldScene.cpp的init()函数中构造 其中MAX\_COUNT为个数

this->map1 = Map<std::string, Sprite \* >();

创建Map类型的map变量 指定Map数据结构中放的是std::string,键为Sprite指针

循环放入元素

for (int i = 0; i < MAX\_COUNT; i++){

\_\_String \* keyBall = \_\_String::createWithFormat("key\_ball\_%d", i);

auto sprite = Sprite::create("CloseSelected.png");

map1.insert(keyBall->getCString(), sprite);

}

在void HelloWorld::menuCloseCallback(Ref\* pSender)中写按钮事件

std::vector<std::string>mapKeyVec;

mapKeyVec = this->map.keys();

for (auto key : mapKeyVec1){

Sprite\* sprite = (Sprite\*)this->map.at(key);

int x = CCRANDOM\_0\_1() \* visibleSize.width;

int y = CCRANDOM\_0\_1() \* visibleSize.height;

sprite->setPosition(Vec2(x, y));

this->removeChild(sprite);

this->addChild(sprite);

}

其中std::vector是标准C++模板类

mapKeyVec是用来保存所有键的数据结构

Sprite\* sprite = (Sprite\*)this->map.at(key); 是根据键获取值对象

~菜单声明

MenuItemFont::setFontName("Times New Roman");

MenuItemFont::setFontSize(86);

MenuItemFont \* item = MenuItemFont::create("Start",

CC\_CALLBACK\_1(HelloWorld::menuItemCallback, this));

其中menuItemCallback需要在HelloWorldScene.h中声明

void menuItemCallback(cocos2d::Ref\* pSender);

在HelloWorldScene.cpp中创建

void HelloWorld::menuItemCallback(Ref\* pSender){

MenuItem\* item = (MenuItem\*)pSender;

}

mn->alignItemsVertically(); 为菜单文字垂直居中

Menu\* mn = Menu::create(item1, item2, NULL); 是创建菜单对象 最后需要用NULL结束

~修改菜单颜色

item->setColor(ccc3(0,0,255)); 前者为菜单指针

~菜单选择时需在HelloWorld.h中加回调声明

void menuItemStartCallBack(cocos2d::Ref\* pSender);

在helloworld.cpp中写入

void HelloWorld::menuItemStartCallBack(cocos2d::Ref\* pSender){}

为创建语句 4个变量 依次正常显示精灵 选择 [禁用] 回调指针

MenuItemSprite\* startMenuItem = MenuItemSprite::create(startNormal,

startSelected,

CC\_CALLBACK\_1(HelloWorld::menuItemStartCallBack, this));

~支持gif格式图片

~vs修改图片 变量 时 名字一致不会重新编译

~开关菜单

文字

auto toggleMenuItem = MenuItemToggle::createWithCallback(

CC\_CALLBACK\_1(HelloWorld::menuItemStartCallBack, this),

MenuItemFont::create("on"),

MenuItemFont::create("off"),

NULL);

Menu\* mu = Menu::create(toggleMenuItem, NULL);

this->addChild(mu);

图片(同上4变量菜单语句)

auto OnMenuItem = MenuItemImage::create(

"Hello.png",

"Hello.png");

auto OffMenuItem = MenuItemImage::create(

"HelloWorld.png",

"HelloWorld.png");

auto Toggle = MenuItemToggle::createWithCallback(

CC\_CALLBACK\_1(HelloWorld::menuItemStartCallBack, this),

OnMenuItem,

OffMenuItem,

NULL);

Toggle->setPosition(Director::getInstance()->convertToGL(Vec2(818, 220)));

Menu\* mn = Menu::create(Toggle,

NULL);

mn->setPosition(Vec2::ZERO);

this->addChild(mn);

~精灵矩形裁剪

auto sprite = Sprite::create("Hello.png",Rect(100,100,200,200));

Rect(); 前两个为坐标 原点在左上角